(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－95772

(43)公開日　平成８年(1996)４月12日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/06	５４０	Ｆ　7230－5Ｂ		

審査請求　未請求　請求項の数６　ＯＬ　（全 12 頁）

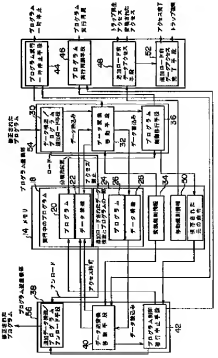
(21)出願番号　特願平6－227613

(22)出願日　平成６年(1994)９月22日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　平山　秀昭
東京都青梅市末広町２丁目９番地　株式会社東芝青梅工場内
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　計算機システム及びプログラム置換方法

(57)【要約】
【目的】システムを停止させることなく、実行中のプログラムの部分的な置換を行なう。
【構成】実行中のプログラムのデータ構造と変更されたプログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段３０と、実行中のプログラムを一時的に停止させるプログラム実行一時停止手段４４と、停止された実行中のプログラムが使用していたデータ構造から追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段３２と、停止された実行中のプログラムと、同プログラムに制御が渡った場合に追加ロードされたプログラムに制御が移るように実行中のプログラムの一部を変更するプログラム制御移行手段３６と、データの移動及びプログラムの変更が完了した後、一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段４６とを具備する。



1

【特許請求の範囲】
【請求項１】　計算機システムにおいて、
実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段と、
前記実行中のプログラムを一時的に停止させるプログラム実行一時停止手段と、
前記プログラム実行一時停止手段によって停止された実行中のプログラムが使用していたデータ構造から、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段と、
前記プログラム実行一時停止手段によって停止された前記実行中のプログラムを、同プログラムに制御が渡った場合に、前記データ構造／プログラム追加ロード手段によって追加ロードされたプログラムに制御が移るように前記実行中のプログラムの一部を変更するプログラム制御移行手段と、
前記データ変換移動手段によるデータの移動、及び前記プログラム制御移行手段によるプログラムの変更が完了した後、前記プログラム実行一時停止手段によって一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段と、
を具備したことを特徴とする計算機システム。
【請求項２】　前記データ構造／プログラム追加ロード手段によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換する追加ロード前データアクセス手段と、
前記追加ロード前データアクセス手段による変換によってアクセスが完了した場合に、トラップから復帰させるための追加ロード前データアクセス完了手段と、
をさらに具備したことを特徴とする請求項１記載の計算機システム。
【請求項３】　前記データ変換移動手段によってデータが移動された追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させるデータ逆変換移動手段と、
前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更するプログラム制御移行中止手段と、
前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造とプログラムをアンロードする追加データ構造／プログラムアンロード手段と、
をさらに具備したことを特徴とする請求項１または請求項２記載の計算機システム。
【請求項４】　計算機システムにおいて、
実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードし、

2

前記実行中のプログラムを一時的に停止し、
停止された実行中のプログラムが使用していたデータ構造から、追加ロードされたデータ構造へデータを変換して移動させ、
前記実行中のプログラムに制御が渡った場合に、前記追加ロードされたプログラムに制御が移るように前記実行中のプログラムの一部を変更し、
前記データの移動、及びプログラムの変更が完了した後、一時的に停止されていたプログラムの実行を再開させることを特徴とするプログラム置換方法。
【請求項５】　前記データの移動、及びプログラムの変更が完了し、一時的に停止されていたプログラムの実行が再開された後、
前記実行中のプログラムの一部によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換し、
この変換によってアクセスが完了した場合に、トラップから復帰させることにより移動されたデータにアクセスされるようにすることを特徴とする請求項４記載のプログラム置換方法。
【請求項６】　前記データの移動、及びプログラムの変更が完了し、一時的に停止されていたプログラムの実行が再開された後、
追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させ、
前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更し、
前記追加ロードされたデータ構造とプログラムをアンロードすることを特徴とする請求項４記載のプログラム置換方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、ソフトウェア面で計算機システムの信頼性を向上させるための、計算機システムのプログラム置換方法に関するものである。
【０００２】
【従来の技術】現在のＯＳ（オペレーティングシステム）、ＤＢＭＳ（データベースマネージメントシステム）、トランザクション処理システム、といった大規模ソフトウェアは、様々な処理要求に対応するため、数百万行の規模に達している。
【０００３】しかし、現在のソフトウェア開発技術は、数百万行の規模のプログラムから、バグを取り除くためのに十分な手法を持ち合わせてはいない。そのため、実際の運用中のプログラムでさえ、バグが内在されたまま、実行が継続されている可能性がある。
【０００４】通常、このようなバグは適時発見され、修

正すべき内容を示すバグ修正情報が作成される。バグ修正情報は、プログラムに適用され、改めてコンパイル／リンクされ、新しい実行モジュールが作成される。

【０００５】従来、バグが修正された新しい実行モジュールは、バグ修正情報適用前の古い実行モジュールの停止を待って、入れ換えが行なわれている。しかしＯＳ、ＤＢＭＳ、トランザクション処理システムといった長時間実行されるシステムでは、実行モジュールの停止の機会が少ないため、実行モジュールの入れ換えの機会もまた少ない。従って、バグ修正情報が作成されたとしても、実際にそのバグ修正情報をプログラムに適用することが困難になっている。

【０００６】今後はシステムの無停止稼働の機会が増え、上記の様な問題は益々深刻となることが予想される。フォールトトレラントシステムによって、ハードウェア故障に起因したシステムダウンは減少させることができたとしても、この様なソフトウェアバグに起因したシステムダウンを減少させることは難しく、しかも処理の高度化と、システムの無停止稼働化に伴って、ソフトウェアバグの問題は益々深刻になってきている。

【０００７】

【発明が解決しようとする課題】このように従来のソフトウェア開発環境においては、プログラムのバグを発見し、バグ修正情報を作成し、コンパイル／リンクして新しい実行モジュールを作成したとしても、バグ修正情報適用前の古い実行モジュールが停止される機会を待って、実行モジュールを入れ換えなければならなかった。すなわち、バグ修正情報のプログラムに対する適用には、システムの停止を伴い、バグを検出した時点で逐次修正することができなかったために、計算機システムの信頼性の向上を迅速に図ることができないという問題があった。

【０００８】本発明は前記のような事情を考慮してなされたもので、システムを停止させることなく、実行中のプログラムの部分的な置換を行なうことが可能なプログラム置換方法を提供することを目的とする。

【０００９】

【課題を解決するための手段】本発明は、実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段と、前記実行中のプログラムを一時的に停止させるプログラム実行一時停止手段と、前記プログラム実行一時停止手段によって停止された実行中のプログラムが使用していたデータ構造から、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段と、前記プログラム実行一時停止手段によって停止された前記実行中のプログラムを、同プログラムに制御が渡った場合に、前記データ構造／プログラム追加ロード手段によって追加ロードされたプログラムに制御が移るように前記実行中のプログラムの一部を変更するプログラム制御移行手段と、前記データ変換移動手段によるデータの移動、及び前記プログラム制御移行手段によるプログラムの変更が完了した後、前記プログラム実行一時停止手段によって一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段とを具備したことを特徴とする。

【００１０】また、前記データ構造／プログラム追加ロード手段によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換する追加ロード前データアクセス手段と、前記追加ロード前データアクセス手段による変換によってアクセスが完了した場合に、トラップから復帰させるための追加ロード前データアクセス完了手段とをさらに具備したことを特徴とする。

【００１１】また、前記データ変換移動手段によってデータが移動された追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させるデータ逆変換移動手段と、前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更するプログラム制御移行中止手段と、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造とプログラムをアンロードする追加データ構造／プログラムアンロード手段とをさらに具備したことを特徴とする。

【００１２】

【作用】このような構成によれば、プログラムのバグを発見し、バグ修正情報を作成し、コンパイル／リンクして新しい実行モジュール（プログラム）を作成した場合に、バグ修正情報適用前の古い実行モジュールの停止、すなわちシステムの停止を待たずに、実行モジュールの入れ換えが実行される。

【００１３】プログラムの入れ換えに伴うデータ構造の変更に対しては、元のデータ構造に対するアクセスがあった場合に、変換後のデータ構造に対するアクセスに変換されるため、実行モジュール（プログラム）と共に、データに関しても入れ換えが実行される。

【００１４】また、バグ修正情報が不適切であった場合等には、バグ修正情報適用前の実行モジュールに制御を戻して、少なくとも前の状態よりも悪くなってしまうことを回避するが、その際にもシステム停止を伴うことが不要とされる。

【００１５】

【実施例】以下、図面を参照して本発明の一実施例を説明する。図１は本実施例に係わる計算機システムの概略構成を示すブロック図である。計算機システム１０には、プロセッサ１２、メモリ１４、及び記憶装置１６が設けられている。

【００１６】メモリ１４には、プロセッサ１２にロードされて実行されるプログラムや各種データ等が格納される。図１においては、本発明によるプログラム置換を行なう前には、実行中のプログラム１８が格納されているものとする。実行中のプログラム１８は、命令列からなるプログラム２０と、命令列によって処理されるデータ構造２２からなるものとする。また、プログラム置換の実行時には、追加ロードされたプログラムとデータ構造の一部２４（追加ロードされたプログラム２６、及び追加ロードされたデータ構造２８からなる）が格納される。

【００１７】また、メモリ１４には、本発明によるプログラム置換機能を実現するプログラム置換プログラムが、例えばＯＳ（オペレーティングシステム）の一部として格納されている。プログラム置換機能については後述する。また、メモリ１４には、プログラム置換に伴って、実行中のプログラムの一部（プログラム置換前の元の命令）が格納される。

【００１８】記憶装置１６は、例えばハードディスク装置等によって構成されるもので、メモリ１４中に格納された実行中のプログラム１８に対応する、ソースプログラム上で更新されコンパイル／リンクされたプログラム１８ａ等が格納されているものとする。

【００１９】図２は、プログラム置換プログラムによって実現されるプログラム置換機能の構成を示すブロック図である。図２に示すように、プログラム置換機能は、データ構造／プログラム追加ロード手段３０、データ変換移動手段３２、プログラム制御移行手段３６、追加データ構造／プログラムアンロード手段３８、データ逆変換移動手段４０、プログラム制御移行中止手段４２、プログラム実行一時停止手段４４、プログラム実行再開手段４６、追加ロード前データアクセス手段４８、及び追加ロード前データアクセス完了手段５２によって構成される。

【００２０】データ構造／プログラム追加ロード手段３０、データ変換移動手段３２、及びプログラム制御移行手段３６は、実行中のプログラムを置換する際に機能するプログラム置換部５４を構成する。

【００２１】また、追加データ構造／プログラムアンロード手段３８、データ逆変換移動手段４０、及びプログラム制御移行中止手段４２は、プログラム置換部５４の各機能によって置換されたプログラムを元の状態に戻すプログラム逆置換部５６を構成する。

【００２２】データ構造／プログラム追加ロード手段３０は、実行中のプログラム１８に対して置換すべきプログラム１８ａをメモリ１４にロード（追加ロードされたプログラム２６、及び追加ロードされたデータ構造２８）するものである。

【００２３】データ変換移動手段３２は、プログラム置換を実行する際に、実行中のプログラム１８のデータ構造２２のデータを、データ構造／プログラム追加ロード手段３０によってメモリ１４に追加ロードされたデータ構造２８に変換して書き込むと共に、実行中のプログラムのデータ構造２８に関してはアクセス禁止状態にするものである。データ変換移動手段３２は、データ構造を変換する際に変換規則に関する情報（変換規則情報３４）を参照し、また変換規則情報３４に基づいて変換して書き込んだデータが存在する位置に関する情報（移動規則情報５０）を作成する。なお、移動規則情報５０は、追加ロード前データアクセス手段４８によってアクセス変換を行なう際に参照されるものである。

【００２４】プログラム制御移行手段３６は、実行中のプログラム１８中の置換されるべきプログラムの命令が実行されず、データ構造／プログラム追加ロード手段３０によってメモリ１４に追加ロードされたプログラムが実行されるように、プログラムの命令の一部を変更するものである。

【００２５】追加データ構造／プログラムアンロード手段３８は、プログラム置換のためにデータ構造／プログラム追加ロード手段３０によって追加ロードされたデータ構造とプログラムの一部２４を、例えば誤りが含まれていたために追加ロードされたプログラム／データ構造の実行が中止された際にアンロードするものである。

【００２６】データ逆変換移動手段４０は、置換されたプログラムを元のプログラムに戻す場合に、追加ロードされたデータ構造２８中のデータを読み込み、元々のデータ構造２２に逆変換して書き込むと共に、元のデータ構造２２がアクセスができるようにアクセス許可状態にするものである。

【００２７】プログラム制御移行中止手段４２は、プログラム置換の際にプログラム制御移行手段３６によって変更された命令を置換前の元の命令に逆変換し、プログラム置換前のプログラムが実行されるようにするものである。

【００２８】プログラム実行一時停止手段４４は、プログラム置換部５４におけるプログラム置換を行なう際、及びプログラム逆置換部５６によってプログラムの逆置換を行なう際に、プロセッサ１２による現在実行中のプログラム実行を一時停止させるものである。

【００２９】プログラム実行再開手段４６は、プログラム実行一時停止手段４４によって現在実行中のプログラム実行が一時停止されている間に、プログラム置換部５４におけるプログラム置換、あるいはプログラム逆置換部５６によってプログラムの逆置換が完了した後、実行中のプログラムの実行を再開させるものである。

【００３０】追加ロード前データアクセス手段４８は、アクセスが禁止されたデータにアクセスする場合に発生されたトラップによって起動され、アクセス禁止されたデータ構造へのアクセスを、追加ロードされ置換されたデータ構造へのアクセスに変換するものである。

7

【００３１】追加ロード前データアクセス完了手段５２は、アクセスが禁止されたデータにアクセスする場合に発生されたトラップを、追加ロード前データアクセス手段４８による変換によってアクセスを終えた後で、トラップから復帰して通常の処理を再開させるものである。

【００３２】次に、本実施例の動作について、図３及び図４に示すフローチャートを参照しながら説明する。まず、実行中のプログラムの一部を置換する動作について、図３を用いて説明する。

【００３３】計算機システム１０では、メモリ１４上にプログラム１８がロードされ、プロセッサ１２によって、そのプログラムを実行している。実行中のプログラム１８は、命令列からなるプログラムと、命令列によって処理されるデータ構造からなる。ここで、実行中のプログラム１８のプログラム上の一部２０と、データ構造の一部２２に、例えばバグが検出されるなどして変更する必要が生じたものとする。

【００３４】そこで、変更すべきプログラムの一部２０と、データ構造の一部２２のみをソースプログラム上で更新し、コンパイル／リンクして得られたプログラム１８ａが記憶装置１６に用意される。

【００３５】データ構造／プログラム追加ロード手段３０は、記憶装置１６に格納された更新されたプログラム１８ａをメモリ１４上に追加ロードする（ステップＡ）。メモリ１４上に追加ロードされたプログラムとデータ構造の一部が２４である。追加ロードされたプログラムとデータ構造の一部２４は、追加ロードされたプログラム２６と、追加ロードされたデータ構造２８を含んでいる。

【００３６】しかし更新したプログラムとデータ構造の一部２４をメモリ１４上に追加ロードしただけでは、実行中のプログラム１８自体は何の影響も受けてなく、プログラムの変更を行なうという事に関しては意味が無い。そこで、以下のようなプログラム置換の処理を実行する。

【００３７】まず、プログラム実行一時停止手段４４は、実行中のプログラム１８の実行を一時停止させる（ステップＡ２）。実行中のプログラム１８が停止されると、データ変換移動手段３２は、実行中のプログラムのデータ構造２２中のデータを読み込み、変換規則情報３４に基づいて、変換後のデータ構造に対応するデータに変換し、追加ロードされたデータ構造２８中に書き込む（ステップＡ３）。そして、実行中のプログラムのデータ構造２２に関してはアクセスができないようにアクセス禁止状態にする。なお、データ構造の変換規則情報３４は、変更すべきプログラムの一部２０と、データ構造の一部２２のみをソースプログラム上で更新し、コンパイル／リンクした際に作成され、プログラム置換を開始する前に、メモリ１４に格納される。変換規則情報３４の詳細については後述する。

8

【００３８】また、プログラム制御移行手段３６は、実行中のプログラム２０のプログラム置換をすべき位置を示すエントリポイントの命令を、追加ロードされたプログラム２６のエントリポイントに無条件分岐する命令に置き換える（ステップＡ４）。また、プログラム制御移行手段３６は、無条件分岐する命令に置換される前の元の命令を保存しておく。

【００３９】これによって、実行中のプログラム１８のプログラムＥとデータ構造２２が変更され、プログラム／データ構造を実行する準備が完了したので、プログラム実行再開手段４６は、実行中のプログラム１８の実行を再開させる（ステップＡ５）。

【００４０】図５にはプログラム置換の具体例を示している。図５（ａ）はプログラム更新前の状態、図５（ｂ）はプログラム更新後の状態を示している。更新前のプログラム２０は、図５（ａ）に示すように、「レジスタの値を保存する」という命令を含む処理ルーチンであり、他の呼び出し側からの要求によって実行され、一連の命令の実行が完了すると呼び出し側に戻るものである。更新前のプログラム２０のエントリポイントの命令は「レジスタの値を保存する」命令となっている。

【００４１】前述したようなプログラム置換を実行すると、図５（ｂ）に示すように、更新前のプログラム２０のエントリポイントの「レジスタの値を保存する」という命令は、更新後のプログラム２６に無条件分岐する命令に置換される。

【００４２】なお、プログラム置換前の命令（「レジスタの値を保存する」）は、予め置換前に保存されている。置換前の命令を保存しておくことにより、後述する更新されたプログラムを元に戻す逆変換の処理を実行する際に、無条件分岐する命令に置換されてしまった更新前のプログラムのエントリポイントの命令を元に戻すことができる。

【００４３】追加ロードされたプログラム２６に制御が移った場合は、通常通りに処理が行なわれる。また、追加ロードされた更新後のプログラム２６によって、更新前のプログラム２０に制御が渡った場合は、エントリポイントの命令が、追加ロードされたプログラム２６のエントリポイントへの無条件分岐命令に置き換えられているので、プログラム２０は実行されず、追加ロードされたプログラム２６に制御が渡る。

【００４４】これによって、プログラム２０に代えてプログラム２６による通常通りの処理が実行される。また、追加ロードされたプログラム２６は、追加ロードされたデータ構造２８にアクセスする場合は、通常通りに処理を実行する。

【００４５】ところで、元の実行中のプログラム１８から、置換されたデータ構造２２にアクセスする場合は、データ構造２２へのアクセスがデータ変換移動手段３２によって禁止されているのでトラップが発生する。

9

【００４６】このトラップによって追加ロード前データアクセス手段４８が呼び出される。追加ロード前データアクセス手段４８は、元のデータ構造２２へのアクセスを、移動規則情報５０に基づいて、追加ロードされたデータ構造２８へのアクセスに変換する。そして、アクセスを終えた後で、追加ロード前データアクセス完了手段５２は、トラップから復帰させて、通常通りの処理を再開させる。なお、移動規則情報５０は、データ変換移動手段３２が追加ロードされたデータ構造２８にデータを変換して書き込む際に作成され、メモリ１４に格納される。移動規則情報５０及びアクセスの変換の詳細については後述する。

【００４７】次に、置換したプログラムの実行を中止して、更新されたプログラムを元に戻す逆変換の動作について、図４に示すフローチャートを参照しながら説明する。例えば、追加ロードしたプログラム２６またはデータ構造２８に誤りが発見された場合等では、プログラム置換を実行する前の状態に戻すことで、少なくとも元のプログラムにおけるバグの影響を越える障害等の発生を回避させることができる。

【００４８】追加ロードされたプログラム２６／データ構造２８の実行を中止したい場合は、まず、プログラム実行一時停止手段４４によって実行中のプログラム１８（追加ロードされたデータ構造とプログラムの一部２４を含む）の実行を一時停止させる（ステップＢ１）。

【００４９】実行中のプログラム１８が停止されると、データ逆変換移動手段４０は、追加ロードされたデータ構造２８中のデータを読み込み、変換規則情報３４に基づいて、元のデータ構造に対応するデータに変換し、元のデータ構造２２に逆変換して書き込む（ステップＢ２）。そして、元のデータ構造２２がアクセスができるようにアクセス許可状態にする。

【００５０】また、プログラム制御移行中止手段４２は、実行中のプログラム２０のエントリポイントの命令を、無条件分岐する命令で置き換える前に保存しておいた元の命令に置き換える（ステップＢ３）。

【００５１】これによって、実行中のプログラム１８のプログラム２０とデータ構造２２が元の状態に戻ったので、プログラム実行再開手段４６は、実行中のプログラム１８の実行を再開させる（ステップＢ４）。

【００５２】追加データ構造／プログラムアンロード手段３８は、メモリ１４に追加ロードしてあった更新後のデータ構造／プログラムを、記憶装置１６にアンロードする（ステップＢ５）。

【００５３】次に、データ構造の変換規則情報の詳細について説明する。変換規則情報は、プログラム置換に伴う更新前のデータ構造と更新後のデータ構造との対応関係を定義するもので、各データ毎に、データタイプとそれらのデータが存在する位置を示すデータ構造の先頭からのオフセットとの組が対応づけられて格納されている。

10

【００５４】図６はデータ構造の変換規則の１番目の例を示す図である。更新前のデータ構造Ａは、図６（ａ）に示すように、ｌｏｎｇ ｉｎｔｅｇｅｒ型のＡ１、ｓｈｏｒｔ ｉｎｔｅｇｅｒ型のＡ２、ｓｈｏｒｔ ｉｎｔｅｇｅｒ型のＡ３から構成されているものとする。なお、図６及び図７に示す例では、例えばｌｏｎｇ ｉｎｔｅｇｅｒ型が４バイト、ｓｈｏｒｔ ｉｎｔｅｇｅｒ型が２バイトであるものとする。

【００５５】また、更新後のデータ構造ａは、図６（ｂ）に示すように、ｌｏｎｇ ｉｎｔｅｇｅｒ型のａ１、ｌｏｎｇ ｉｎｔｅｇｅｒ型のａ２、ｌｏｎｇ ｉｎｔｅｇｅｒ型のａ３から構成されているものとする。

【００５６】この時、変換規則情報Ａａは、図６（ｃ）に示すような関係で定義される。すなわち、データ構造Ａのオフセット０からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数は、データ構造ａのオフセット０からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数に対応し、同様にしてデータ構造Ａのオフセット４からのｓｈｏｒｔ ｉｎｔｅｇｅｒ型の変数はデータ構造ａのオフセット４からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数に対応し、データ構造Ａのオフセット６からのｓｈｏｒｔ ｉｎｔｅｇｅｒ型の変数はデータ構造ａのオフセット８からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数に対応している。

【００５７】図７はデータ構造の変換規則の２番目の例を示す図である。更新前のデータ構造Ｂは、図７（ａ）に示すように、ｌｏｎｇ ｉｎｔｅｇｅｒ型のＢ１、ｌｏｎｇ ｉｎｔｅｇｅｒ型のＢ２から構成されているものとする。

【００５８】また、更新後のデータ構造ｂは、図７（ｂ）に示すように、ｌｏｎｇ ｉｎｔｅｇｅｒ型のｂ１、ｌｏｎｇ ｉｎｔｅｇｅｒ型のｂ２、ｌｏｎｇ ｉｎｔｅｇｅｒ型のｂ３から構成されているものとする。

【００５９】この時、変換規則ＢＢは、図７（ｃ）に示すような関係で定義される。すなわち、データ構造Ｂのオフセット０からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数はデータ構造ｂのオフセット０からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数に対応し、データ構造Ｂのオフセット４からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数はデータ構造ｂのオフセット４からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数に対応し、データ構造ｂのオフセット８からのｌｏｎｇ ｉｎｔｅｇｅｒ型の変数は新たに追加されている。

【００６０】データ変換移動手段３２は、図６（ｃ）または図７（ｃ）に示すような変換規則情報に基づいて、実行中のプログラム１８のデータ構造２２のデータを、追加ロードされたデータ構造２８に合わせて変換して書き込むことができる。

【００６１】また、データ逆変換移動手段４０は、同様にして変換規則情報に基づいて、追加ロードされたデー

11

タ構造２８のデータを、元のデータ構造２２に合わせて変換して書き込むことができるので、プログラム置換の前の元の状態にすることができる。

【００６２】なお、図６及び図７では、ｉｎｔｅｇｅｒ型のデータ型についての み示しているが、他のデータ型に応用することも可能である。次に、移動規則情報、及び追加ロードされたデータ構造へのアクセス（アクセスの変換）について詳細を説明する。移動規則情報は、プログラム置換に伴う更新前のデータ構造と更新後のデータ構造のそれぞれにおけるデータの位置の対応関係を定義するもので、データが位置する開始番地とそれらのデータ長との組が対応づけられて格納されている。

【００６３】図８は追加ロードされたデータ構造へのアクセスの例を示す図である。なお、図８ではページ単位でアクセス制御されるものとする。また、アクセス制御は、例えばページ毎に設けられた保護ビットに“１”をセットすることにより、アクセス禁止にする方式がとられる。

【００６４】図８（ａ）は更新前のデータ構造（データ構造２２）を含むページＰで、プログラム置換に伴ってアクセス禁止状態にされているものとする。このページＰの中にはＸ番地にｌｏｎｇ　ｉｎｔｅｇｅｒ型の変数からなるデータ構造Ａが存在し、Ｙ番地に２つのｓｈｏｒｔ　ｉｎｔｅｇｅｒ型の変数からなるデータ構造Ｂが存在している。

【００６５】図８（ｂ）は更新後のデータ構造（データ構造２８）を含むページＱで、アクセス可能状態になっている。このページＱの中にはＺ番地に２つのｌｏｎｇ　ｉｎｔｅｇｅｒ型の変数からなるデータ構造ｂが存在している。データ構造ｂは、図８（ａ）中のデータ構造Ｂを更新したものである。

【００６６】ページＰはアクセス禁止状態になっているので、このページ上に存在する全てのデータ構造へのアクセスにおいてトラップが発生する。トラップが発生した場合は、追加ロード前データアクセス手段４８が呼び出される。追加ロード前データアクセス手段４８は、トラップ処理の中で、図８（ｃ）に示すような移動規則情報Ｒを検査する。

【００６７】追加ロード前データアクセス手段４８は、移動規則情報Ｒを参照して、トラップが発生した場合に、そのページ上のデータ構造にアクセスすべきか、あるいはデータ構造が更新されているため他のデータ構造にアクセスすべきかを判定することができる。

【００６８】例えば、ページＰのＸ番地のデータ構造Ａにアクセスしてトラップが発生した場合、Ｘ番地からのデータ構造Ａが移動規則Ｒに登録されていないので、追加ロード前データアクセス手段４８は、ページＰ上のＸ番地のデータ構造Ａにアクセスさせる。

【００６９】一方、Ｙ番地のデータ構造Ｂにアクセスしてトラップが発生した場合、Ｙ番地からのデータ構造Ｂ

12

が移動規則Ｒに登録されているので、追加ロード前データアクセス手段４８は、ページＰ上のＹ番地のデータ構造Ｂの代わりに、ページＱ上にＺ番地のデータ構造ｂにアクセスさせる。

【００７０】こうして、移動規則情報に応じてアクセスを変換するので、プログラム置換を行なう際に、追加ロードされたデータ構造２８に対しても正しくアクセスすることができる。

【００７１】このようにして、実行中のプログラム１８にバグが検出された場合に、バグ修正情報を作成し、このバグ修正情報が適用されたプログラムを改めてコンパイル／リンクして新しい実行モジュールを作成した場合に、実行中のプログラムが停止されるシステム停止の機会を待つことなく、適宜、プログラム置換機能による追加ロードされたプログラム２６及びデータ構造２８の置換によって修正することができる。

【００７２】追加ロードされたデータ構造２８に対しては、追加ロードされる前の元のデータ構造２２に対するアクセスがあった場合にトラップを発生させて、トラップ処理の なかでアクセス変換を行なうことによって、アクセスできるように制御される。

【００７３】また、追加ロードしたプログラム２６またはデータ構造２８に誤りが発見された場合でも、システム停止の機会を待つことなく、バグ修正情報適用前の状態に制御を戻すことができる。

【００７４】

【発明の効果】以上詳述したように本発明によれば、システムを停止させることなく、実行中のプログラムの部分的な置換を行なうことが可能となるものである。

【図面の簡単な説明】

【図１】本発明の一実施例に係わる計算機システムの概略構成を示すブロック図。

【図２】本発明によるプログラム置換機能の構成を示すブロック図。

【図３】本実施例におけるプログラム置換の動作を説明するためのフローチャート。

【図４】本実施例におけるプログラム逆変換の動作を説明するためのフローチャート。

【図５】本実施例におけるプログラム置換の具体例を示す図。

【図６】本実施例におけるデータ構造の変換規則の１番目の例を示す図。

【図７】本実施例におけるデータ構造の変換規則の２番目の例を示す図。

【図８】本実施例における追加ロードされたデータ構造へのアクセス（アクセスの変換）を説明するための図。

【符号の説明】

１０…計算機システム、１２…プロセッサ、１４…メモリ、１６…記憶装置、１８…実行中のプログラム、２０，２６…プログラム、２２，２８…データ構造、２４

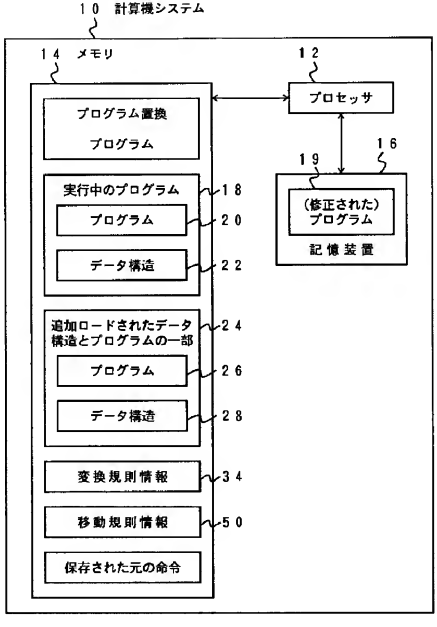
13

…追加ロードされたプログラムとデータ構造の一部、３０…データ構造／プログラム追加ロード手段、３２…データ変換移動手段、３４…変換規則情報、３６…プログラム制御移行手段、３８…追加データ構造／プログラムアンロード手段、４０…データ逆変換移動手段、４２…

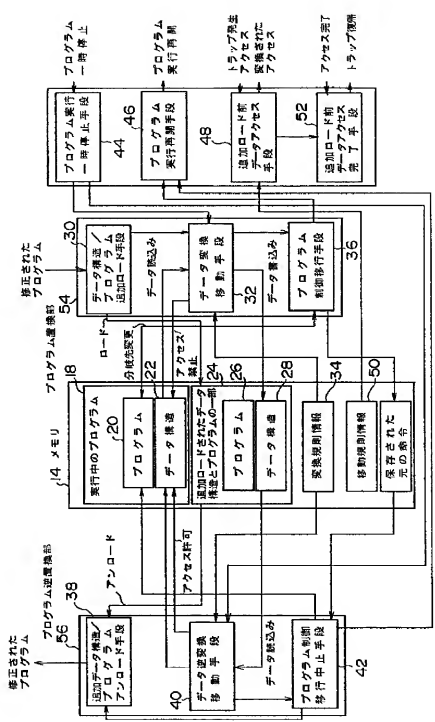
14

プログラム制御移行中止手段、４４…プログラム実行一時停止手段、４６…プログラム実行再開手段、４８…追加ロード前データアクセス手段、５０…移動規則情報、５２…追加ロード前データアクセス完了手段、５４…プログラム置換部、５６…プログラム逆置換部。

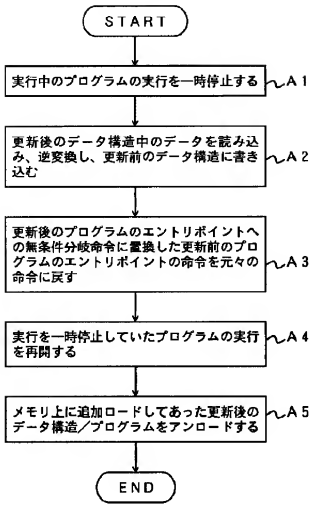
【図１】



【図2】


プログラム一時停止
プログラム実行再開
トラップ発生アクセス変換されたアクセス
アクセス完了トラップ復帰
プログラム実行一時停止手段
44
プログラム実行再開手段
46
48
追加ロード前データアクセス手段
52
追加ロード前データアクセス完了手段
修正されたプログラム
プログラム置換部
54
30
データ構造／プログラム追加ロード手段
データ読込み
データ変換移動手段
32
データ書込み
プログラム転送実行手段
36
ロード
分岐先変更
アクセス禁止
18
22
24
26
28
34
50
14 メモリ
実行中のプログラム
20
プログラム
データ構造
追加ロードされたデータ構造とプログラムの一部
プログラム
データ構造
変換規則情報
移動規則情報
保存された元の命令
アクセス許可
アンロード
プログラム逆置換部
38
修正されたプログラム
56
追加データ構造／プログラムアンロード手段
データ変換移動手段
40
データ読込み
プログラム制御移行中止手段
42

【図3】



【図6】

(a) 更新前のデータ構造A

A 1 (long)	
A 2 (short)	A 3 (short)

(b) 更新後のデータ構造 a

a 1 (long)
a 2 (long)
a 3 (long)

(c) 変換説明情報A a

データ構造A		データ構造 a	
オフセット	タイプ	オフセット	タイプ
＋0	long	＋0	long
＋4	short	＋4	long
＋6	short	＋8	long

【図7】

(a) 更新前のデータ構造B

B 1 (long)
B 2 (long)

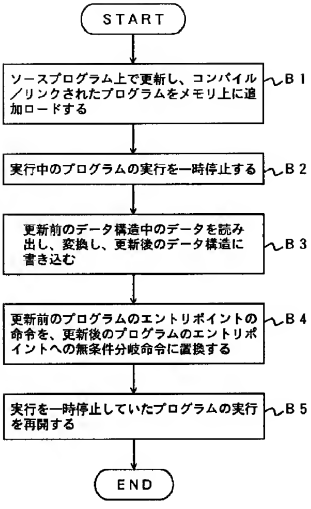
(b) 更新後のデータ構造 b

b 1 (long)
b 2 (long)
b 3 (long)

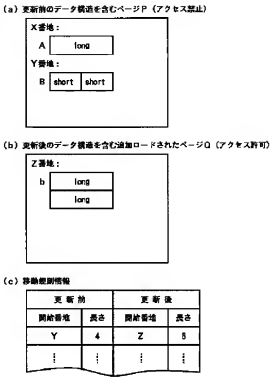
(c) 変換説明情報

データ構造B		データ構造 b	
オフセット	タイプ	オフセット	タイプ
＋0	long	＋0	long
＋4	long	＋4	long
－	－	＋8	long

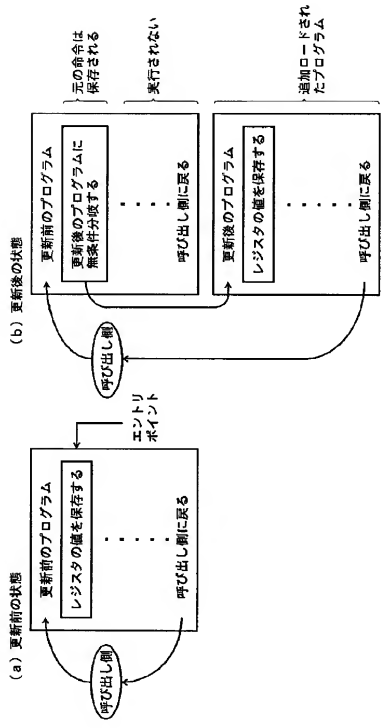
【図4】


START
ソースプログラム上で更新し、コンパイル／リンクされたプログラムをメモリ上に追加ロードする
B1
実行中のプログラムの実行を一時停止する
B2
更新前のデータ構造中のデータを読み出し、変換し、更新後のデータ構造に書き込む
B3
更新前のプログラムのエントリポイントの命令を、更新後のプログラムのエントリポイントへの無条件分岐命令に置換する
B4
実行を一時停止していたプログラムの実行を再開する
B5
END

【図8】


(a) 更新前のデータ構造を含むページP（アクセス禁止）
X番地：
A
long
Y番地：
B
short
short
(b) 更新後のデータ構造を含む追加ロードされたページQ（アクセス許可）
Z番地：
b
long
long
(c) 移動経路情報
更新前
更新後
開始番地
長さ
開始番地
長さ
Y
4
Z
5

【図5】


(a) 更新前の状態
更新前のプログラム
レジスタの値を保存する
呼び出し側に戻る
呼び出し側
エントリポイント
(b) 更新後の状態
更新前のプログラム
更新後のプログラムに無条件分岐する
呼び出し側に戻る
元の命令は保存される
実行されない
更新後のプログラム
レジスタの値を保存する
呼び出し側に戻る
追加ロードされたプログラム